Usando Funções de Similaridade para Redução de Conjuntos de Casos de Teste em Estratégias de Teste Baseado em Modelos

Emanuela G. Cartaxo[1], Patrícia D. L. Machado[1], Francisco G. Oliveira Neto[1], João F. S. Ouriques[1]

[1] GES/DSC, Universidade Federal de Campina Grande (UFCG), Brazil

{emanuela,patricia,netojin,jfelipe}@dsc.ufcg.edu.br

Resumo. *O uso de funções de similaridade para a redução de conjuntos de casos de teste é investigado neste trabalho. É dado enfoque a conjuntos de casos de teste funcionais gerados automaticamente a partir de modelos. No geral, estratégias de geração automática de casos de teste produzem conjuntos muito grandes e com muitos casos de teste redundantes, contribuindo para um aumento significativo e desnecessário nos custos de geração e execução de casos de teste. Redução com base em funções de similaridade visa produzir um conjunto com um número menor de casos de teste que contenha os casos de teste menos similares entre si. Com isto, espera-se obter uma cobertura mais efetiva de funcionalidades. As estratégias de similaridade consideradas são comparadas, através de estudos de caso, com seleção aleatória, com base em dois critérios: cobertura de transições e cobertura de defeitos.*

Abstract. *The use of similarity functions for test suite reduction is investigated in this work. The focus is on automatically generated functional test cases from models. In general, test case generation strategies produce large test suites with a considerable number of redundant test cases, causing an unnecessary and non-significant increase in the test case generation and execution costs. The main goal of reduction based on similarity functions is to produce a smaller test suite that has the less similar test cases with a more effective coverage of functionalities. The considered similarity strategies are compared with random selection, according to two criteria: transition coverage and fault coverage.*

1. Introdução

Um dos grandes desafios para a adoção de estratégias de teste na indústria é a produção de conjuntos de casos teste que sejam efetivos quer seja na detecção de defeitos ou na validação de requisitos de um software e, ao mesmo tempo, utilizar recursos disponíveis de forma otimizada, comprometendo minimamente o orçamento destinado à produção do software. Neste sentido, pesquisas na área de teste de software têm sido cada vez mais orientadas para vencer este desafio [Grindal et al. 2006, Bertolino 2007].

Teste baseado em modelos (*Model Based Testing* - MBT) é uma abordagem para teste funcional de software na qual conjuntos de casos de teste são derivados, possivelmente de forma automática, a partir de um modelo que representa o comportamento desejado para o software a ser testado [El-Far and Whittaker 2001]. MBT tem sido apontada como uma abordagem promissora para aumentar a confiabilidade, efetividade e produtividade em processos de teste, visto que promete controlar a qualidade de software e também

reduzir os custos inerentes [Pretschner 2005]. Com MBT, casos de teste podem ser obtidos assim que a especificação estiver pronta e, quando o software estiver implementado, os casos de teste já poderão ser executados.

Uma das limitações de MBT é que a quantidade de casos de teste gerados é usualmente muito grande, e nem sempre há recursos disponíveis para executar o conjunto completo. Assim, é necessário selecionar apenas alguns casos de teste para serem executados. Contudo, esta pode ser uma tarefa muito difícil, visto que é necessário selecionar um subconjunto representativo que assegure tanto critérios estruturais quanto critérios funcionais. Outro ponto observado em conjuntos gerados automaticamente é a grande quantidade de casos de teste redundantes. Neste artigo, consideramos que um caso de teste é reduntante se: i) não aumenta significativamente a capacidade do conjunto em atingir uma melhor cobertura de um certo critério; ou ii) não aumenta a capacidade do conjunto em revelar defeitos ainda não cobertos. Por fim, é importante observar que, devido à automação, casos de teste gerados tendem a ser maiores, com relação ao número de passos para sua execução, do que casos de teste escritos manualmente e comumente executados na prática. Casos de teste grandes são mais difíceis e caros para executar, mais sujeitos a erros na execução manual e podem dificultar a localização de defeitos [Jorgensen 2002].

Apesar de já existirem abordagens voltadas a redução de conjuntos de teste, na prática, a seleção de casos de teste e, consequentemente, a redução de conjuntos de teste tem sido, no geral, executada de forma empírica e manual. Desta forma, a seleção pode ficar mais sujeita a erros e não nos dar garantia de que a aplicação será testada apropriadamente, uma vez que critérios como cobertura ou detecção de defeitos não são tomados como parâmetros e sistematicamente aplicados.

Este trabalho tem como enfoque o uso de funções de similaridade como parte de uma estratégia de seleção de casos de teste que é aplicável a abordagens de MBT. A idéia é selecionar os casos de teste menos similares, considerando as seguintes hipóteses: i) casos de teste similares são redundantes, uma vez que cobrem um mesmo conjunto de funcionalidades e têm a mesma capacidade de revelar defeitos; ii) não há ganhos relativos em manter casos de teste similares e sua eliminação pode contribuir para a redução de custos do processo de teste.

A estratégia é voltada para sistemas de transições rotuladas (*Labelled Transition Systems* - LTSs) como modelos de onde os casos de teste podem ser gerados. Neste caso, os casos de teste são caminhos selecionados através do caminhamento sobre o modelo do estado inicial ao final. LTSs não são comumente usados na prática para modelar aplicações, mas são adotados como a semântica de vários formalismos de especificação. Um número considerável de ferramentas têm sido desenvolvidas para suportar a geração de casos de teste partindo de LTSs que são derivados de modelos abstratos tais como UML, a exemplo das ferramentas TGV [Jard and Jéron 2005, Hartman and Nagin 2004] e LTS-BT [Cartaxo et al. 2008]. Assim, a estratégia aqui apresentada pode ser integrada com diferentes abordagens de MBT e ferramentas. Dado um LTS e uma porcentagem de cobertura de caminhos, a estratégia reduz a quantidade de casos de teste, assegurando que os casos de teste que permanecem no conjunto são os mais diferentes possíveis.

A estratégia apresentada neste artigo foi inicialmente introduzida por Cartaxo et al., com experimentos preliminares comparando sua efetividade com a seleção aleatória,

considerando a cobertura de transições [Cartaxo et al. 2007]. A contribuição original deste artigo refere-se a proposta e investigação de uma abordagem da estratégia de similaridade, onde os casos de teste com menor tamanho são priorizados e sua comparação experimental com a abordagem original, que prioriza os casos de teste com maior tamanho, e com seleção aleatória, considerando critérios de comparação como cobertura de transições e de defeitos revelados pelos casos de teste.

Este artigo está estruturado da seguinte forma. A Seção 2 apresenta conceitos básicos sobre MBT e mostra uma definição formal de LTSs. Na Seção 3, a estratégia de redução de conjuntos de casos de teste baseada em similaridade é apresentada. Na Seção 4, são apresentados e discutidos os resultados da aplicação da estratégia em três estudos de caso. A Seção 5 apresenta os trabalhos relacionados. Por fim, na Seção 6 são apresentadas considerações finais sobre os resultados obtidos.

2. Teste e Modelos

Esta seção apresenta conceitos básicos de MBT através de uma breve descrição de um modelo de processo genérico. Em seguida, o modelo de testes abordado neste trabalho, LTS, é definido formalmente.

2.1. MBT

MBT é uma abordagem de teste que faz uso de modelos, tipicamente chamados de modelos de teste, para representar o comportamento que se deseja avaliar em uma implementação. A Figura 1 mostra um processo típico de MBT com as seguintes fases:

1. **Construção do Modelo.** Nesta fase, uma representação mental dos requisitos de um sistema é construída e documentada como um modelo. O objetivo é produzir um modelo voltado para teste (*test ready model* [Binder 2000]) que contém informações suficientes para tornar possível a geração automática de casos de teste. Modelos de desenvolvimento e documentos de requisitos podem ser usados como um ponto de partida. Contudo, modelos de teste usualmente precisam ser mais detalhados e completos.
2. **Geração de Casos de Teste.** Casos de teste são derivados a partir do modelo. A automação desta fase depende de qualidade do modelo: modelos formais usualmente tornam possível a automação. Algoritmos para geração de casos de teste são, no geral, baseados em critérios de cobertura estrutural (todos os caminhos, todos os nós, dentre outros) sobre os modelos. Casos de teste são gerados a partir de caminhamentos sobre o modelo até que um certo conjunto de critérios seja atingido. Estratégias para redução de conjuntos de teste, tais como abordada neste artigo, devem ser combinadas com algoritmos de geração para a produção de conjuntos com um número mínimo de casos de teste. Este combinação é particularmente relevante quando critérios semânticos são considerados, tais como similaridade, que permitam distinguir os casos de teste.
3. **Execução de Casos de Teste.** Consiste em executar conjuntos de casos de teste gerados previamente.
4. **Coleta e Análise de Resultados.** Resultados da execução de testes e métricas relacionadas, tais como cobertura e tempo de execução, são coletadas e analisadas para apoiar a tomada de decisões com relação a: i) casos de teste são suficientes; ii) mais casos de teste são necessários; iii) modelo precisa ser aprimorado.

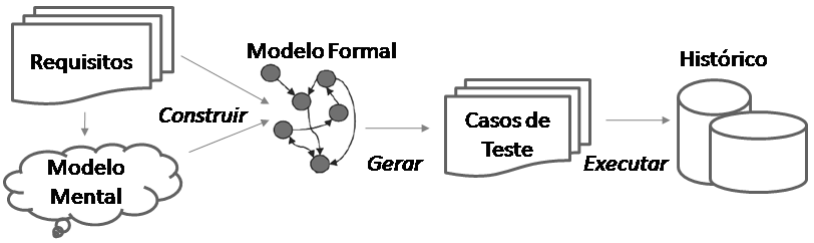

Figura 1. Um Modelo de Processo de MBT[Nascimento and Machado 2007].

2.2. Sistema de Transições Rotuladas

Sistema de Transições Rotuladas (*Labelled Transition System* - LTS) é um modelo formal usado para representar o comportamento de uma dada aplicação. LTSs são amplamente utilizados como formalismo semântico de muitas notações de especificação [Jard and Jéron 2005] e, por isso, podem ser facilmente obtidos de especificações funcionais usando ferramentas de conversão, como por exemplo UMLAUT [Ho et al. 1999]. Além disso, LTSs também são amplamente utilizados como o modelo do qual os casos de teste podem ser obtidos, por exemplo, nas ferramentas SPACES [Barbosa et al. 2007], TGV [Jard and Jéron 2005], LTS-BT [Cartaxo et al. 2008] e TaRGeT [Nogueira et al. 2007]. LTS é um grafo dirigido - dígrafo - no qual vértices são chamados de estados e arestas de transições. Estas são rotuladas por letras de um alfabeto ou eventos. Podemos definir formalmente um LTS como uma 4-tupla $S = (Q, A, T, q_0)$, onde [de Vries and Tretmans 1998]:

- Q é um conjunto não vazio e finito de estados;
- A é um conjunto não vazio e finito de rótulos (que denotam ações);
- T, a relação de transição, é um subconjunto de $Q \times A \times Q$;
- q_0, com q_0 pertencendo a Q, é o estado inicial.

Os modelos LTS que são usados na prática para a geração de casos de teste são baseados em variantes desta definição. Por exemplo, Sistemas de Transições Rotuladas com Entrada e Saída (*Input Output Labelled Transition Systems* - IOLTS), que é o modelo usado pela ferramenta TGV, distingue entradas, saídas e ações internas no conjunto A. Por outro lado, os Sistemas de Transições Rotuladas Anotadas (*Annotated Labelled Transition Systems* - ALTS), o modelo utilizado pelas ferramentas LTS-BT e TaRGeT, consideram que os rótulos podem indicar ações de entrada, resultados esperados, condições associadas às entradas e também anotações para distinguí-las e guiar a geração dos casos de teste.

3. Redução Baseada em Similaridade

Nesta seção, é apresentada uma estratégia de redução de caso de testes. Esta estratégia é baseada na remoção de casos de testes com informações redundantes, medidas através de grau de similaridade. Este trabalho foi inspirado no trabalho de Jin e Yeh [Lin and Yeh 2001] que gera dados de teste usando algoritmos genéticos e para isto é calculada uma função de avaliação baseada na distância de Hamming (diferença simétrica de um conjunto).

Para aplicar esta estratégia, é necessário que o testador especifique o modelo comportamental LTS, bem como uma porcentagem de caminhos que devem ser cobertos,

considerando que cada caso de teste é um caminho no LTS. O algoritmo de geração de casos de teste percorre o modelo de teste a partir do estado inicial utilizando uma busca em profundidade (*Depth First Search*) a fim de obter todos os caminhos do modelo. Como resultado, o conjunto de todos os possíveis casos de testes é selecionado do modelo.

Para reduzir a quantidade de casos de teste, é estabelecido um grau de similaridade entre cada par de casos de teste, de forma que um dos dois casos de testes com maior grau de similaridade será removido. Dentre os dois casos de teste com maior similaridade, duas abordagens podem ser aplicadas para a escolha do caso de teste a ser removido (conseqüentemente, o caso de teste a ser mantido):

1. Original - É escolhido para ser removido o caso de teste com o menor número de transições. Com isto, o objetivo é favorecer a melhor cobertura de funcionalidades, visto que os casos de teste com maior número de passos são mantidos.
2. Menor Caso de Teste - É escolhido para ser removido o caso de teste com o maior número de transições. O objetivo é produzir um conjunto mais simples (menores casos de teste).

Este processo é iterativo, e ocorre até que a porcentagem de cobertura de caminhos (quantidade de casos de teste) especificada pelo testador seja atingida. Para aplicar este processo, é necessário que, inicialmente, seja construída uma matriz de similaridade da seguinte forma:

- N $\times$ N, onde N é o número de casos de teste obtidos a partir dos caminhos do modelo LTS;
- Cada elemento a_{ij} da matriz representa a similaridade entre os casos de testes i e j determinada pela função $FunçãoSimilaridade(i, j)$ calculada a partir do número de transições iguais (nti) entre os casos de teste (transições que possuem os mesmos estados origem e destino e o mesmo rótulo) e a média entre os caminhos do caso de teste ($med(|i|, |j|)$).

Definindo formalmente (considerando um LTS), consideramos que duas transições são iguais se,

$$\forall q \xrightarrow{\alpha} q' \epsilon\, T \; e \; q'' \xrightarrow{\alpha'} q''' \epsilon\, T \;, q = q'' \; e \; q' = q''' \; e \; \alpha = \alpha'$$

$$FunçãoSimilaridade(i, j) = \frac{nti}{med(|i|, |j|)} \tag{1}$$

É importante observar a natureza simétrica da matriz, já que a similaridade entre os casos de teste i e j, é a mesma quando se considera o par j e i ($a_{ij} = a_{ji}$). Dessa forma, é desnecessário considerar o cálculo do grau de similaridade para os elementos da matriz onde $i = j$.

Dentre os elementos da $FunçãoSimilaridade(i, j)$ especificada, o nti é utilizado para observar as funcionalidades semelhantes entre dois casos de teste, enquanto que a divisão pela média dos tamanhos dos casos de teste é necessária para ponderar a similaridade, evitando, dessa forma, que a matriz apresente uma pequena similaridade entre um par de casos de teste com pequeno tamanho, e uma grande similaridade entre um par de casos de teste de tamanho grande que não sejam muito similares.

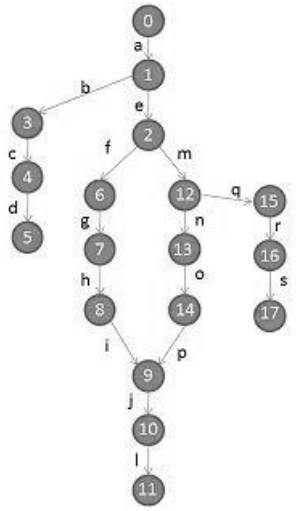

Figura 2. Exemplo de um modelo comportamental LTS.

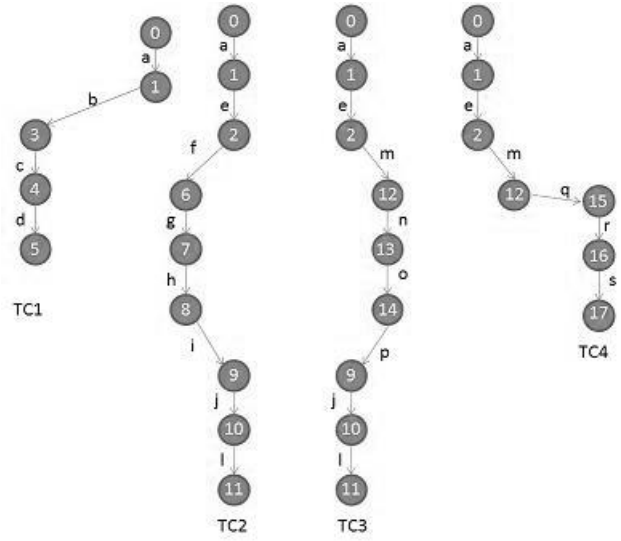

Figura 3. Casos de teste gerados a partir do modelo da Figura 2.

Com o objetivo de ilustrar a abordagem original da estratégia, é apresentado a seguir um exemplo de aplicação da estratégia, considerando o modelo comportamental da Figura 2.

Executando uma busca em profundidade (*Deph First Search* - DFS) no modelo, é possível gerar 4 casos de teste. Considerando que é necessário reduzir a quantidade de casos de teste devido à escassez dos recursos disponíveis para a execução dos casos de teste, é preciso estabelecer quais casos de teste selecionar, de acordo com a porcentagem de cobertura de caminhos definida. Pode-se observar na Figura 3 os casos de teste gerados a partir do modelo especificado, e na Figura 4 (a) e (b), a matriz de similaridade dos casos de teste e o tamanho de cada um destes respectivamente.

Considerando que a porcentagem especificada pelo testador para a execução deste cenário é 50% dos casos de teste gerados, é necessário remover 2 dos 4 casos de teste gerados. Para isto, é observado o grau de similaridade dos casos de teste na matriz de

Figura 4. - (a) Matriz de Similaridade, (b) Comprimento dos caminhos, (c) Matriz de similaridade após remoção.

similaridade construída. Como se pode observar na Figura 4 (a), o maior grau de similaridade da matriz é 0.5, representando a similaridade entre os casos de teste TC2 e TC3, calculado da seguinte forma:

- Número de transições iguais (nti): 4;
- Média entre o comprimento dos caminhos ($med(|i|, |j|)$): 8;
- Grau de Similaridade: $\frac{4}{8} = 0,5$.

Uma vez obtido o maior grau de similaridade da matriz (0,5), é necessário estabelecer qual dos dois casos de teste (TC2 e TC3) remover. Para isto, observa-se o número de transições presentes nos dois casos de teste. Já que os dois casos de teste possuem o mesmo tamanho (quantidade de transições), uma escolha aleatória é feita para definir qual dos dois casos de teste remover. Considere que a escolha resultou na remoção de TC2.

Com a remoção de TC2, a linha e coluna do caso de teste removido são também removidas da matriz, resultando na matriz da Figura 4(c). O processo é repetido, pois ainda é necessário remover 1 caso de teste, dentre os remanescentes. Da mesma forma que antes, identifica-se o maior grau de similaridade na matriz. Neste caso, o maior grau de similaridade é 0,42 e é entre TC3 e TC4. Como $|\text{TC3}| > |\text{TC4}|$, TC4 é removido de acordo com a abordagem original. Com o percentual especificado atingido, os casos de teste restantes são TC1 e TC3. É importante observar que os casos de teste removidos (TC2 e TC4) são os mais similares aos casos de testes que permaneceram no conjunto de teste (TC1 e TC3, respectivamente).

No exemplo apresentado, temos apenas 4 casos de teste gerados, que possivelmente poderiam ser executados. No entanto, para um modelo comportamental LTS de uma aplicação real, a quantidade de casos de teste gerada é usualmente muito grande e, em projetos reais, com restrições de tempo e recursos, 100% de cobertura pode ser inviável, sendo necessário selecionar um subconjunto representativo dos casos de teste.

Por fim, é importante ressaltar que a estratégia de redução baseada em similaridade apresentada nesta seção é automatizada pelas ferramentas LTS-BT [Cartaxo et al. 2008] e TaRGeT [Nogueira et al. 2007].

4. Avaliando a estratégia

Para avaliar a estratégia de similaridade, foram executados três estudos de caso. Estes consistiram na geração e redução automáticas de casos de teste usando as duas abordagens da estratégia de similaridade (original e menor caso de teste) e também seleção aleatória

de casos de teste. A seleção aleatória consiste na escolha aleatória de um caso de teste por vez, a ser removido do conjunto de casos de teste com base em uma função aleatória com distribuição uniforme de probabilidades. A motivação para a escolha de seleção aleatória é que esta estratégia tem se mostrado mais efetiva na prática do que escolha determinística com relação à cobertura e diversidade de escolhas. Portanto, compatível com as expectativas definidas para as estratégias de similaridade.

Para comparar as estratégias, foram considerados dois critérios:

- Cobertura de transições - O número total de transições que cada estratégia cobre ao considerar todos os casos de teste selecionados. A idéia é observar se as estratégias preservam cobertura de transições;
- Cobertura de defeitos - O número total de defeitos que são revelados pelo conjunto de casos de testes durante a execução. Para isto, foram definidos modelos de defeitos para os estudos de caso e foram consideradas versões dos estudos de caso com defeitos detectáveis pelos casos de teste. A idéia é observar o quanto as estratégias preservam a capacidade de detecção de defeitos do conjunto original.

Estes critérios foram escolhidos com o fim de responder a questões tais como: (i) Escolha com base em função de similaridade é mais efetiva do que a escolha aleatória? (ii) Quais os limites em cada estratégia? Qual é mais ou menos efetiva? (iii) A esperada perda de cobertura de transições com a estratégia do menor caso de teste também se reflete em perda da cobertura de defeitos? O quão significativa é esta perda? (iv) Em que circunstâncias é mais recomendável adotar cada uma das estratégias?

Como dito anteriormente para cada estudo de caso, as três estratégias foram aplicadas. Para cada estratégia, a porcentagem de cobertura de caminhos variou de 5% a 95% (incrementada de 5 em 5). Além disto, devido ao fator aleatório que existe nas três estratégias, para cada porcentagem de cobertura de caminhos, a estratégia foi executada 100 vezes e os valores considerados para cada porcentagem foi definido como a média obtida com as 100 execuções. O objetivo é identificar qual das estratégias (listadas acima) assegura a melhor redução do conjunto, baseado nos critérios levantados.

Para a condução dos estudos de caso, foi utilizada a ferramenta LTS-BT para aplicar as estratégias de similaridade, bem como a geração do conjunto completo de casos de teste de forma automática.

Nas próximas seções, as aplicações relativas aos três estudos de caso são brevemente descritas. Em seguida, são apresentados os resultados obtidos.

4.1. Estudos de Caso - Descrição

As três aplicações escolhidas (referenciadas como Estudos de Caso 1, 2 e 3) são reativas, ou seja, aplicações que reagem ao estímulo de seu ambiente. Para os Estudos de Caso 1 e 2, temos aplicações para celulares, e para o estudo de caso 3, temos uma aplicação para *desktop*. Resumidamente, temos:

- Estudo de caso 1 – Aplicação para adicionar contatos na lista de contatos de um celular;
- Estudo de caso 2 – Aplicação que lida com itens embutidos (URL, número de telefone ou email) de uma mensagem, uma vez que para cada item embutido pode-se ter várias ações;

- Estudo de caso 3 – Aplicação que gera casos de teste automaticamene a partir de cenários de uso - ferramenta TaRGeT [Nogueira et al. 2007].

Para os três estudos de caso, o mesmo processo foi seguido para obter o modelo comportamental LTS e foi conduzido pela mesma equipe. Os modelos obtidos cobrem 100% dos requisitos conhecidos de cada aplicação.

A Tabela 1 mostra, para cada estudo de caso, o número casos de teste, de transições e defeitos, considerando 100% de cobertura de caminhos. Para o Estudo de Caso 3, foram considerados defeitos reais detectados pela execução da aplicação. Já para os Estudos de Caso 1 e 2, foram definidos modelos de defeitos. Tais modelos contém defeitos que podem ser incluídos na implementação, dada uma interpretação errônea por parte do programador. Desta forma, a implementação produz respostas que não estão em conformidade com o modelo LTS.

Tabela 1. Estudos de Caso - Métricas

Estudo de Caso	Casos de Teste	Transições	Defeitos
1	24	121	6
2	66	826	8
3	130	631	13

A Tabela 2 apresenta a taxa de defeitos por número de transições e casos de teste e também a taxa de casos de teste por transições que caracteriza o grau de similaridade de caminhos na aplicação e, conseqüentemente, o grau de similaridade dos casos de teste.

Tabela 2. Estudos de Caso - Defeitos por número de transições e casos de teste, e casos de teste por transições (Taxa de Similaridade)

	Estudo de Caso 1	Estudo de Caso 2	Estudo de Caso 3
Defeitos / Número de Transições	0,050	0,010	0,021
Defeitos / Casos de Testes	0,250	0,121	0,100
Casos de teste / Transições	0,198	0,080	0,206

A Tabela 3 mostra dados sobre os tamanhos dos casos de teste de cada estudo de caso. Note que o Estudo de Caso 1 possui mais casos de teste com tamanho acima do tamanho médio (moda e média com valor alto relativo ao maior e menor caso de teste), enquanto que o Estudo de Caso 3 possui mais casos de teste com tamanho abaixo do tamanho médio (moda e média com valor baixo relativo ao maior e menor caso de teste). No Estudo de Caso 2, prevalecem casos de teste com tamanho médio.

Tabela 3. Estudos de Caso - Dados sobre o tamanho dos casos de teste (quantidade de transições).

	Estudo de Caso 1	Estudo de Caso 2	Estudo de Caso 3
Menor Caso de Teste	6,0	11,0	6,0
Maior Caso de Teste	29,0	27,0	38,0
Tamanho Médio	17,5	19,0	22,0
Moda	24,0	19,0	14,0
Média	22,17	18,06	20,93

Neste artigo, por questões de confidencialidade e simplicidade, os modelos LTS dos três estudos de caso não são apresentados. Entretanto, é importante ressaltar que os

Estudos de Caso 1 e 3 têm mais casos de teste redundantes que o Estudo de Caso 2. Tal fato, pode também ser concluído ao se observar a taxa de casos de teste por transições na Tabela 2.

4.2. Estudos de Caso - Resultados

Nesta seção, são apresentados e discutidos os resultados obtidos, considerando cobertura de transições (Subseção 4.2.1) e de defeitos (Subseção 4.2.2) para cada aplicação.

4.2.1. Cobertura de Transições

No modelo LTS de cada aplicação, as funcionalidades são representadas como transições rotuladas. Portanto, ao considerar este critério, foram contadas as transições cobertas pelo conjunto reduzido de casos de teste para cada estratégia. As Figuras 5, 6 e 7 representam os resultados obtidos para os Estudos de Caso 1, 2 e 3, respectivamente, quando aplicamos cada uma das três estratégias.

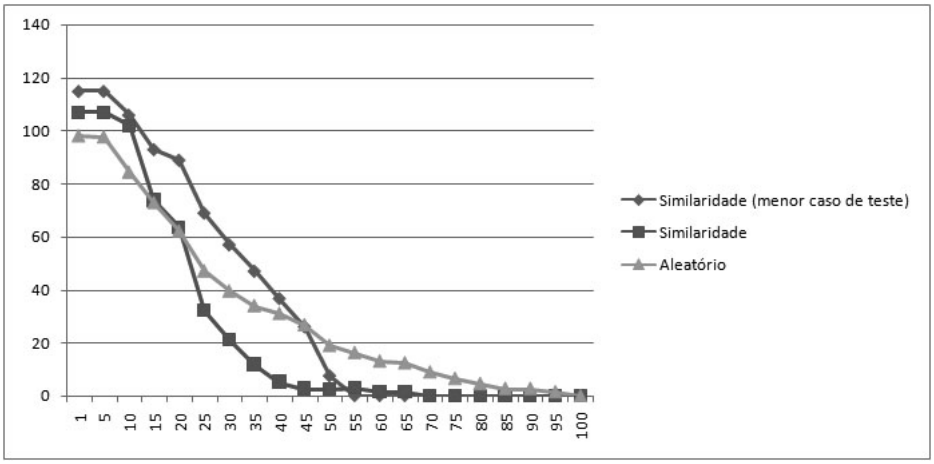

Figura 5. Número médio de transições excluídas ao executar cada estratégia de seleção 100 vezes para cada porcentagem - Estudo de Caso 1

Para cada uma das figuras, foi representado no eixo das abcissas (eixo x) a porcentagem de cobertura de caminhos, variando de 5 a 95%. No eixo das ordenadas (eixo y), foi representada a média de transições excluídas, considerando cada uma das estratégias.

Como mencionado anteriormente, os Estudos de Caso 1 e 3 apresentam mais casos de teste redundantes. Já o Estudo de Caso 2 apresenta similaridade entre pares de casos de teste. É notório que a estratégia original de similaridade é mais efetiva para o Estudo de Caso 2. Independente da porcentagem de cobertura de caminhos, ele escolhe os caminhos mais diferentes e garante a melhor cobertura de transições. Comparando os resultados obtidos, estes sugerem que geralmente para cobertura de caminhos de até 20%, é mais indicado utilizar a estratégia de similaridade original.

Com relação a estratégia do menor caso de teste, como era esperado, esta obviamente perde em corbertura para a estratégia original (que favorece os maiores casos de teste). Portanto, é relevante apenas observar sua efetividade com relação a estratégia aleatória. Neste caso, pode-se observar que, para o Estudo de Caso 1, até 50% de redução,

a estratégia aleatória se mostra mais interessante. Este estudo de caso é o que possui uma maior população de casos de teste com tamanho acima da média (ver Tabela 3). O ganho acima de 50% é obtido pela habilidade da estratégia de similaridade em capturar os casos de teste mais diferentes. Para o Estudo de Caso 2, onde prevalecem casos de teste de tamanho médio, as estratégias possuem resultados comparáveis, com um pequeno ganho para a aleatória. Por fim, para o Estudo de Caso 3, as estratégias têm um desempenho bastante semelhante (as variações são devido às escolhas aleatórias). Neste estudo de caso, prevalecem casos de teste com tamanho abaixo da média (ver Tabela 3).

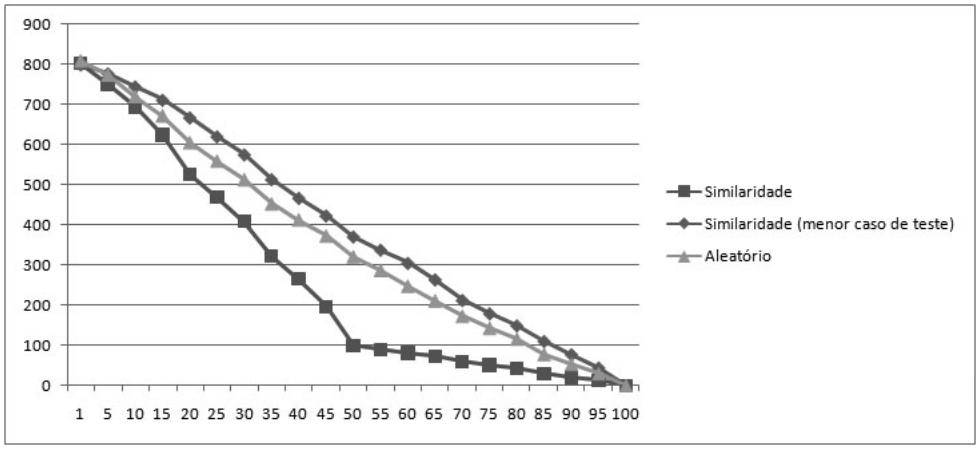

Figura 6. Número médio de transições excluídas ao executar cada estratégia de seleção 100 vezes para cada porcentagem - Estudo de Caso 2

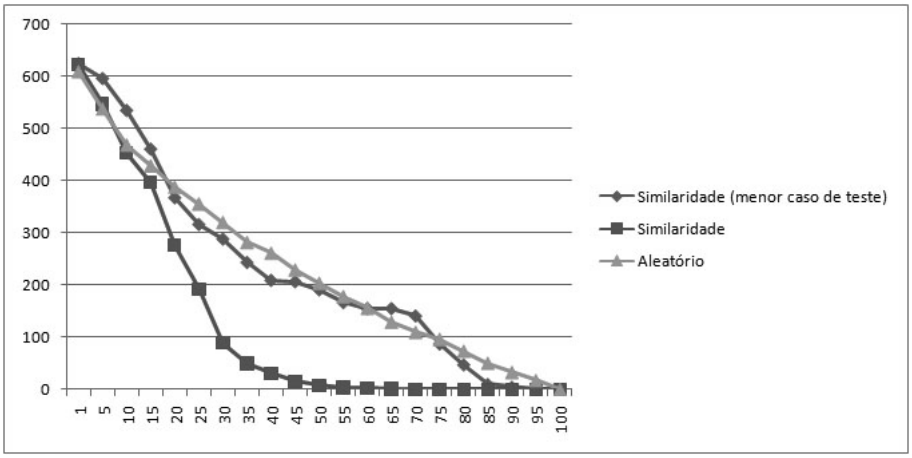

Figura 7. Número médio de transições excluídas ao executar cada estratégia de seleção 100 vezes para cada porcentagem - Estudo de Caso 3

4.2.2. Cobertura de Defeitos

Para cada estudo de caso, casos de teste foram associados com os defeitos que são capazes de revelar. Assim, o número de defeitos cobertos por um conjunto de casos de teste é computado através do número total de diferentes defeitos cobertos por aqueles casos de teste. As Figuras 8, 9 e 10 apresentam os resultados obtidos.

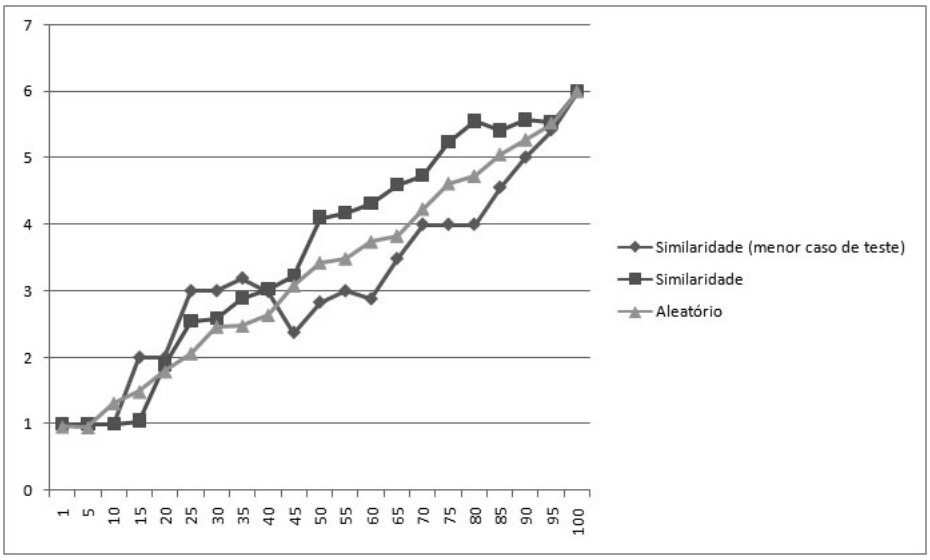

Figura 8. Número médio de defeitos descobertos ao executar cada estratégia de seleção 100 vezes para cada porcentagem - Estudo de Caso 1

Para cada uma das figuras, foi representado, no eixo das abcissas, a porcentagem de cobertura de caminhos, variando de 5 a 95%. No eixo das ordenadas, foi representada a média de defeitos cobertos considerando cada uma das estratégias.

Comparando os resultados obtidos, a estratégia original de similaridade se mostra mais vantajosa, apenas com ressalvas para o Estudo de Caso 1, onde supera a aleatória apenas para cobertura acima de 20%, e a similaridade com menor caso de teste a partir de 40%. Esta vantagem sobre a aleatória é explicada pelo fato de que a estratégia de similaridade mantém os casos de teste mais diferentes, sendo portanto mais efetiva na seleção de conjuntos que preservam a capacidade de detecção de defeitos do conjunto original. Já a vantagem da original sobre a menor caso de teste, e vice-versa, está diretamente relacionada a distribuição de defeitos entre casos de teste grandes e pequenos. Como o Estudo de Caso 1 tem menos casos de teste pequenos, é possível concluir que estes concentram o maior número de defeitos.

Com relação a estratégia do menor caso de teste, é possível notar que sua aplicação é vantajosa apenas no Estudo de Caso 3, onde predominam casos de teste de tamanho pequeno.

É importante comentar que os resultados obtidos nestes experimentos podem ter sido influenciados pela taxa de defeitos por casos de teste e o tamanho da aplicação (medido aqui pelo número de transições). A Tabela 2 sumariza estes resultados. Para o Estudo de Caso 1, houve um ganho menos significativo da estratégia de similaridade. Isto pode ser explicado pela maior taxa de defeitos e casos de teste por transições que pode maximizar as chances da estratégia aleatória fazer boas escolhas. Já o Estudo de Caso 3, com menor taxa de defeitos por casos de teste, apresentou os melhores resultados para similaridade.

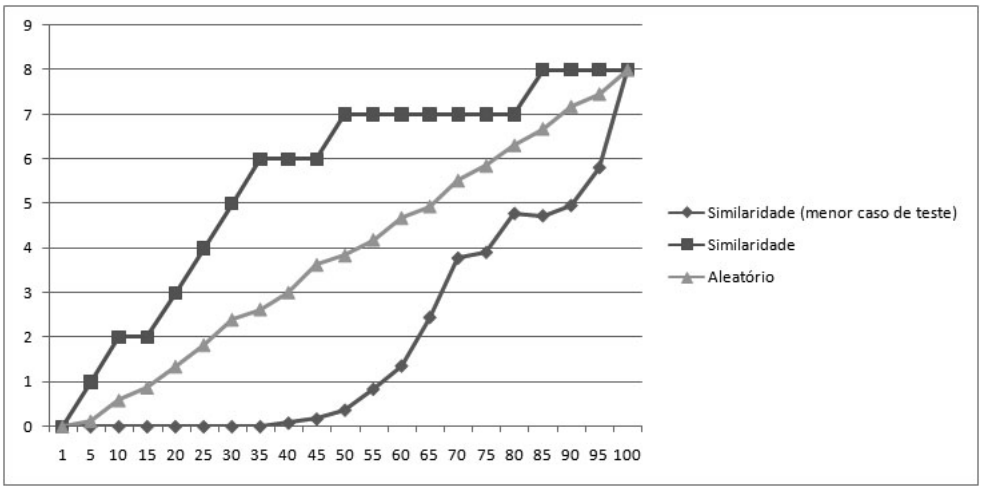

Figura 9. Número médio de defeitos descobertos ao executar cada estratégia de seleção 100 vezes para cada porcentagem - Estudo de Caso 2

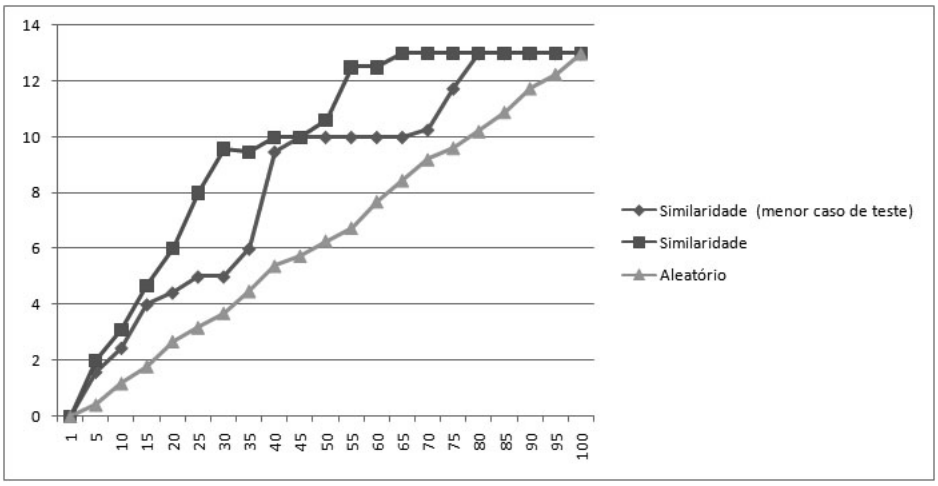

Figura 10. Número médio de defeitos descobertos ao executar cada estratégia de seleção 100 vezes para cada porcentagem - Estudo de Caso 3

4.2.3. Comentários Gerais

Independente das variações, os resultados obtidos com os estudos de caso sugerem que a estratégia de similaridade, particularmente a original, é mais efetiva do que a aleatória ao considerar uma cobertura de caminhos maior que 20%. Esta vantagem abaixo de 20% para a aleatória no Estudo de Caso 1 pode ser explicada pelo fato de que possui um número de defeitos relativamente maior do que as outras aplicações e casos de teste similares (ver Tabela 2). Com relação as abordagens de similaridade original e com menor caso de teste, pode-se observar que as vantagens esperadas para a primeira devido a maior cobertura de transições podem não ser tão significativas se: i) casos de teste pequenos cobrem grande parte dos defeitos; ii) casos de teste pequenos são predominantes.

5. Trabalhos Relacionados

Harrold et al. apresentou uma metodologia para a redução do tamanho de conjuntos de casos de teste, onde a entrada é uma matriz de rastreabilidade de requisitos [Harrold et al. 1993]. A matriz é composta por requisitos e casos de teste, por exemplo, para cada requisito, a matriz apresenta os casos de teste que o cobre. Aplicando essa metodologia, é possível obter 100% de cobertura dos requisitos com um conjunto reduzido de casos de teste. Todavia, a metodologia exposta não verifica a possível redundância nos casos de teste que estão cobrindo os requisitos, nem a cobertura de transições.

Para especificações UML, como os diagramas de seqüência e diagramas de caso de uso, temos a ferramenta Cow Suite [Basanieri et al. 2002]. Nela, a seleção de casos de teste é feita considerando uma função de peso, por exemplo, para cada diagrama considerado, é atribuído um peso estimado da importância funcional. A fim de aplicar essa estratégia, é necessário ter um testador experiente para atribuir corretamente os pesos. Também usando modelos UML temos a ferramenta SPACES [Barbosa et al. 2007]. Ela é usada para teste funcional de componentes. Para cada transição do modelo, um peso é atribuído. De acordo com esses pesos, o conjunto de casos de teste mais importante é selecionado. A nossa estratégia pode ser integrada em ambas as ferramentas para tratar a redundância entre os casos de teste como uma estratégia de redução do conjunto de casos de testes.

Uma avaliação de estratégias para seleção de casos de teste é apresentada (*All Combination Strategy, Each Choice Strategy, Base Choice Strategy, Automatic Efficient Test Generator e Orthogonal Arrays*) [Grindal et al. 2006]. Essas estratégias são mais dedicadas às atividades de teste onde um número de combinações entre parâmetros e valores precisa ser considerado. Novamente, nossa estratégia pode ser aplicada para reduzir o conjunto de testes produzida por elas, sempre que aplicável: quantas das combinações selecionadas podem continuar sendo redundantes?

Já Mello e Simão apresentam um algoritmo de minimização de conjuntos de testes para Máquinas de Estado Finitas (MEF) baseado em condições de suficiência [de Mello Neto and Simão 2007]. Essas condições garantem que quando um conjunto de casos de teste as satisfazem, então esse conjunto de casos de teste é n-completo. O algoritmo apresentado, reduz um conjunto de casos de teste de modo que o conjunto de casos de teste final também seja n-completo, garantindo cobertura completa de erros na MEF. Porém este algoritmo não pode ser aplicado quando o conjunto inicial não é n-completo ou quando ele é n-completo e não atende as condições de suficiência.

Por fim, Wong et al. apresentou um estudo empírico com o objetivo de avaliar os efeitos da redução do tamanho de conjuntos de testes, sempre mantendo a cobertura de bloco [Wong et al. 1999]. A conclusão desse trabalho é que conjuntos representativos têm quase que a mesma capacidade de revelar defeitos de que conjunto original. Zhong *et al* apresentou um estudo comparativo entre quatro técnicas de redução do conjunto de testes: heurística H [Harrold et al. 1993], heurística GRE [Chen and Lau 1998], abordagem baseada em algoritmos genéticos e uma abordagem baseada em ILP [Zhong et al. 2006]. Os fatores considerados nessa comparação são conjuntos representativos e tempo de execução.

6. Considerações Finais

Este artigo apresentou uma estratégia para seleção de casos de teste baseada no uso de função de similaridade, cujo objetivo é reduzir um conjunto de casos de teste, mantendo apenas os mais diferentes. Duas abordagens foram ressaltadas: uma voltada a manter o maior caso de teste no conjunto, dentre dois casos de teste similares, e a outra voltada a manter o menor caso de teste. Com o objetivo de avaliar a estratégia e suas abordagens, foram conduzidos três estudos de caso, onde a estratégia foi comparada a seleção aleatória, com base em dois critérios de cobertura (defeitos e transições). A estratégia aleatória se mostrou mais vantajosa apenas em um dos estudos de caso, quando a redução é relativa a percentuais menores ou iguais que 20%. Em todos os outros casos, similaridade é claramente bem mais vantajosa: a estratégia pode ser usada para reduzir a redundância dos conjuntos e manter uma cobertura aceitável dos critérios investigados com relação ao conjunto original. Dado que a entrada da estratégia é um LTS, ela pode ser largamente aplicada na prática e adotada por diferentes ferramentas. Atualmente duas ferramentas implementam a estratégia apresentada: LTS-BT [Cartaxo et al. 2008] e TaRGeT [Nogueira et al. 2007].

Como trabalhos futuros, é importante investigar outras funções de similaridade que, por exemplo, levem em consideração a importância relativa dos casos de teste e defeitos. Outros experimentos precisam ser conduzidos para investigar o impacto da redução considerando outros critérios, tais como tempo de execução e cobertura de requisitos.

Referências

Barbosa, D. L., Lima, H. S., Machado, P. D. L., Figueiredo, J. C. A., Jucá, M. A., and Andrade, W. L. (2007). Automating functional testing of components from uml specifications. *Int. Journal of Software Eng. and Knowledge Engineering*, 17:339–358.

Basanieri, F., Bertolino, A., and Marchetti, E. (2002). The cow suite approach to planning and deriving test suites in uml projects. In *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 383–397, London, UK. Springer-Verlag.

Bertolino, A. (2007). Software testing research: Achievements, challenges, dreams. In *Proceedings of FOSE'07: 2007 Future of Software Engineering - ICSE*, pages 85–103. IEEE Computer Society.

Binder, R. V. (2000). *Testing Object-Oriented Systems - Models, Patterns and Tools*. Addison-Wesley.

Cartaxo, E. G., Andrade, W. L., Neto, F. G. O., and Machado, P. D. L. (2008). LTS-BT: a tool to generate and select functional test cases for embedded systems. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, volume 2, pages 1540–1544, New York, NY, USA. ACM.

Cartaxo, E. G., de Oliveira Neto, F. G., and Machado, P. D. L. (2007). Automated test case selection based on a similarity function. In *Proceedings of MOTES07 - Model-based Testing - Workshop in conjunction with the 37th Annual Congress of the Gesellschaft fuer Informatik*, volume 110 of *Lecture Notes in Informatics (LNI)*, pages 381–386.

Chen, T. and Lau, M. (1998). A new heuristic for test suite reduction. *Information and Software Technology*, 40(5):347–354.

de Mello Neto, L. F. and Simão, A. (2007). Minimização de conjuntos de casos de teste por meio de condições de suficiência. In *1st Brazilian Workshop on Systematic and Automated Software Testing*, João Pessoa, PB, Brazil. SBBD-SBES.

de Vries, R. G. and Tretmans, J. (1998). On-the-fly conformance testing using SPIN. In *Proceedings of Fourth Workshop on Automata Theoretic Verification with the Spin Model Checker*, pages 115–128.

El-Far, I. K. and Whittaker, J. A. (2001). Model-based software testing. *Encyclopedia on Software Engineering*.

Grindal, M., Lindström, B., Offutt, J., and Andler, S. F. (2006). An evaluation of combination strategies for test case selection. *Empirical Softw. Eng.*, 11(4):583–611.

Harrold, M. J., Gupta, R., and Soffa, M. L. (1993). A methodology for controlling the size of a test suite. *ACM Trans. Softw. Eng. Methodol.*, 2(3):270–285.

Hartman, A. and Nagin, K. (2004). The agedis tools for model based testing. *SIGSOFT Softw. Eng. Notes*, 29(4):129–132.

Ho, W. M., Jézéquel, J.-M., Guennec, A. L., and Pennaneac'h, F. (1999). UMLAUT: An extendible uml transformation framework. In *ASE '99: Proceedings of the 14th IEEE international conference on Automated software engineering*, Washington, DC, USA. IEEE Computer Society.

Jard, C. and Jéron, T. (2005). Tgv: theory, principles and algorithms: A tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems. *Int. J. Softw. Tools Technol. Transf.*, 7(4):297–315.

Jorgensen, P. C. (2002). *Software Testing - A Craftsman's Approach*. CRC Press.

Lin, J.-C. and Yeh, P.-L. (2001). Automatic test data generation for path testing using GAs. *Inf. Sci.*, 131(1-4):47–64.

Nascimento, L. H. O. and Machado, P. D. L. (2007). An experimental evaluation of approaches to feature testing in the mobile phone applications domain. In *Proceedings of Domain-Specific Approaches to Software Test Automation (DoSTA) - Satellite workshop of ESEC/FSE 2007*, pages 27–33.

Nogueira, S., Cartaxo, E., Torres, D., Aranha, E., and Marques, R. (2007). Model based test generation: An industrial experience. In *1st Brazilian Workshop on Systematic and Automated Software Testing - SBBD/SBES 2007*, João Pessoa, PB, Brazil.

Pretschner, A. (2005). Model-based testing. In *Proceedings of International Conference on Software Engineering - ICSE*, pages 722–723.

Wong, W. E., Horgan, J. R., Mathur, A. P., and Pasquini, A. (1999). Test set size minimization and fault detection effectiveness: a case study in a space application. *J. Syst. Softw.*, 48(2):79–89.

Zhong, H., Zhang, L., and Mei, H. (2006). An experimental comparison of four test suite reduction techniques. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 636–640, New York, NY, USA. ACM.